Leakage Detection in Waterpipes Networks using Acoustic Sensors and Identifying Codes

Francesco De Pellegrini and Roberto Riggio
CREATE-NET
Via alla Cascata 56/D Povo, 38123
Trento - Italy
Email: {fdepellegrini,rriggio}@create-net.org

***Abstract*—In future smart cities a grand challenge will be to sensorize large urban infrastructures at a feasible cost. In this paper we tackle the case of efficient leakage detection in water distribution systems. Deploying leakage detectors can cut operational costs for water utility providers. But, the cost for deploying them with sufficient granularity poses an high entrance barrier due to the scale of such infrastructures. We propose an algorithmic framework to efficiently deploy sensors in order to perform leakage/fault localization over large scale lattice–shaped networks. The novelty of our solution, combining covering sets and identifying codes is that it initially covers the network with low resolution, and thus fewer sensors. The set of sensors can then be extended in a way to progressively improve the resolution by which leakages are located. The proposed solution is validated through extensive numerical experiments.**

***Index Terms*—Leakage detection, waterpipes networks, sensor placement, covering, identifying codes.**

I. Introduction

The usage of sensors promises major savings in the operational costs for many industrial and urban infrastructures. In this paper, we focus on the specific case of the metropolitan water distribution systems (WDS). In such systems, the infrastructure's aging process cause ever increasing operational costs. Water pipes are in fact constantly under pressure, with seasonal variations that may as well depend on the demand. Overall, the effect is the constant aging of the pipes and the formation of leakages in the infrastructure.

However, repairing the leakage is not always the most convenient strategy. To do so, in fact, the exact position of a leakage has first to be determined. Unfortunately, utility providers often know only the aggregated water demand of a certain district. Thus, one can not distinguish a water leak from ordinary customers' demands. In order to discriminate such events, real–time water meters at the customer ends should be deployed. Such a solution raises many technical issues for wiring and powering such devices and is not viable nowadays.

The current practice to detect the leakages is to measure the *Minimum Night flow (MNF)* [1]. Since the demand is typically negligible at that time, the water consumed during night hours provides an upper bound to the leakages in a certain district. Major breaches trigger major variations of MNFs and are thus relatively easy to locate, e.g., due to surface flooding.

This work has been partially supported by the European project OUTSMART FP7-2011-ICT-FI- 285038.

Less severe leakages, conversely, may be difficult to spot and thus pose to the utility provider the dilemma whether to search and repair the leakage, or to delay the repair and continue serving the customers facing a constant increased cost due to pressure overprovisiong in the waterpipes.

A customary technique to identify the position of a leakage in a water pipe is performed by placing two synchronized acoustic correlators at the end points of the leaking pipe. Such devices allow the maintenance crew to precisely identify the position of the leakage. To make such an operation economically viable, though, one should restrict the area of intervention as much as possible, i.e., to identify a small subset of waterpipes, that can be potential hosts for leakages.

In this paper we propose a technique for the localization of leakages which aims at minimizing the set of candidate waterpipes. We leverage on a class of noise loggers capable, once placed at wells or at junctions of water pipes, to identify the presence of a leakage within a certain range. The problem thus becomes the following: *given a certain number of sensors, we want to deploy them is such a way to minimize the number of pipes that are candidate to host a leakage.* To do so, we formulate leakage detection as a covering problem on a graph where edges are the water pipes and nodes are junctions: the sensor deployment must detect the noise generated by a leakage located in any point of the network. In the ideal case, we want to precisely identify uniquely the pipe, e.g., the edge of the graph, that hosts a leakage.

Moreover, we want the solution to be incremental: we wish to start with a bootstrap coverage configuration where if any leakage exists, it will be detected. But, we can add progressively sensors until a configuration which minimize the number of candidate pipes hosting the leakage, i.e. an identifying set of sensors, is obtained. The rationale is dictated by economic considerations: the initial deployment would be less accurate, but would permit to sanitize the network with few sensors. Then, after the network is sanitized, the provider could progressively deploy more sensors which could identify new leakages with greater confidence. Finally, it is worth noticing that such problem is indeed relevant for other type of piped distribution infrastructures, e.g. oil, gas or electricity.

Related works

The way we solve the problem is by identifying vertices in graphs: such codes were proposed in the seminal work of Karpovsky et al. [2]. Variants of the identification problem include the case when one node covers more than one edge. The case when the distance of the identifying code is larger than one was introduced first in [3].

Traditional link–based monitoring schemes for computers networks require that a dedicated monitoring facility is available at each link; this requires $O(|E|)$ monitors, where $|E|$ is the number of links in the network. *M-trail* is an active path probing solution proposed in [4] localizing single failures in a computer network in more scalable fashion. In [5], a solution based on several m-trails per routing entity solves optimally fault localization for lattice networks. In general, trail-based approaches, such as companion m-trees [6] and m-cycles [7] apply well for communication infrastructure, but they do not apply to passive infrastructures such as waterpipes networks.

In [8] the authors optimize the placement of sensors in municipal water networks to detect the injection of contaminants. An integer programming formulation is proposed; simulations demonstrate that the resulting sensor configuration is relatively insensitive to uncertainties in the data used for prediction. Similarly, authors of [9] developed multi-objective optimization for water distribution systems: they minimize for the expected water volume contaminated and the expected detection time and maximizing for the detection likelihood. Optimization is performed using a genetic algorithm.

Our goal, is different compared to those works, i.e., we aim at minimizing the number of water pipes to be manually probed in case of leakages.

The rest of the paper is structured as follows. The leakage detection problem together with the network model is introduced in Sec. II. Section III formulates the problem. An iterative puncturing algorithm which minimizes the number of sensors required to identify leakages is introduced in Sec. IV. Section V tackles the incremental identification problem, while Sec. VI presents numerical results. Finally, Sec. VII draws the conclusions pointing out some interesting future research directions.

II. Leakage detection in pipelines

The requirements for this work are dictated by the water leakage sensing technologies currently available on the market. We are considering a particular class of sensors suitable for large scale deployment called *acoustic loggers*: such sensors are deployed at junctions or wells along waterpipes to detect the noise generated by water leakages. Acoustic loggers periodically measure the noise level on the waterpipe and the analysis of this data allows the logger to detect the presence of a leakage. The event of leakage detection is transmitted to the network manager using various networking technologies; M2M commercial devices such as the Primayer Phocus.SMS[1] deliver an SMS over the cellular network. The alarm can include the noise strength and the confidence factor, the latter represents the likelihood that a leak is actually present within the coverage radius of the sensor (see Table I).

[1]Online resources available at: http://www.primayer.co.uk/

Factor	Meaning and confidence	Noise Strength
2	Leak noise – confindent/very confident.	$\geq$ 41dB
1	Possibly leak noise + other noise	31 - 41 dB
0	No leak noise	$\leq$ 30dB

TABLE I: *Noise strength scale.*

The waterpipe network is represented as an euclidean undirected graph, i.e., a graph $G = (V, E)$ with N vertices which represent either a waterpipe junction or a well, and M edges $e_1, \dots, e_M$ where vertices lie on a plane; we consider the origin of the plane fixed and refer to any point of the plane as the position vector. $v_i \in V$ will indicate both the vertex label of the i-th node and its position. In particular, we are interested in points that correspond to certain positions along edges of G, which we denote briefly as $x_i \in e_j$ to say that x_i lies on edge e_j. Such positions will be denoted $X = \{x_1, \dots, x_L\}$: as it will be clear in the following, X represents the training set for our XCODE algorithm (see Sec.V).

We can define $\ell(x_i, v_j) = |x_i - v_j|$ as the (geometric) distance of x_j from v_j along the edges of graph G. Element $x_j \in X$ produces a signature at node v_i according to a characteristic function, i.e., $h(\cdot)$. Given $\ell' = \ell(x_i, v_j)$, the value $h(\ell')$ represents the strength registered at v_j of a leakage signal emitted at x_i. In our study we assume that the signal characteristic of a leakage $h(\ell)$ is a decreasing threshold function of the distance ℓ.

$$h(\ell) = a_k, \quad \ell \in [t_{k-1}, t_k), k = 1, \dots, K \tag{1}$$

where tresholds $0 = t_0 \leq t_1 \leq \dots \leq t_K$ and $h(t) = 0$ for $t \geq t_K$. Expression (1) generalizes the noise strength scale reported in Table I. In our study we refer to the case where $K = 2$ for simplicity's sake. Moreover we confine our model to the case when $h(\cdot)$ does not depend on the materials and junctions encountered along the path by the traveling sound wave. We leave those aspects as part of future work.

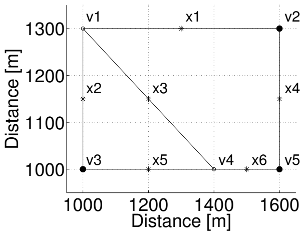
Define $\mathbb{C} \subseteq V$, $\mathbb{C} = \{v_{j_1}, \dots, v_{j_L}\}$. Consider $x_i \in X$: and let $\ell_{ij_r} = \ell(x_i, v_{j_r})$: hence it is possible to define the $|X|$ codewords of length L: $c_{x_i} = (h(\ell_{ij_1}), \dots, h(\ell_{ij_L}))$, $i = 1, \dots, |X|$. $\mathcal{C}(\mathbb{C})$ is the set of codewords for (G, X).

With standard coding terminology, the distance of codewords h_1 and h_2 is the number of positions where they differ, i.e., $d(c_1, c_2) := \sum_{i=1}^{N} \ ((c_1)_i \neq (c_2)_i)$. Denote $\mathbb{C}$ a $[n, m, d]$ if it is composed of words of length n (namely, the code length), has m codewords and the code distance is d, where code distance $d(\mathcal{C}) = \min\{d(c_i, c_j), c_i \neq c_j \in \mathbb{C}\}$

We say that $\mathbb{C}$ is an *X-identifying set* for (G, X) if *code* $\mathcal{C} = \mathcal{C}(\mathbb{C})$ has distance $d \geq 1$.

Also, $0 \in \mathcal{C}(\mathbb{C})$ since the empty syndrome is always a legitimate codeword (no leakage detected).

The i-punctured code $\mathcal{C}_i^*$ is defined as the code that is obtained removing the i-th component from all codewords. The punctured code has length which is shortened by one compared to the original length.



$$\mathcal{C}(\mathbb{C})=\begin{array}{c}\begin{pmatrix} 0 & 1 & 2 & 0 & 2 \\ 0 & 2 & 1 & 0 & 2 \\ 0 & 2 & 2 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \end{pmatrix}\end{array}\begin{array}{l} \to x_1 \\ \to x_2 \\ \to x_3 \\ \to x_4 \\ \to x_5 \\ \to x_6 \end{array}$$

Fig. 1: *An example of a code built for graph G and set X given by mid-edge points;* $t_1 = 500$ *m and* $t_2 = 800$ *m.*

Puncturing a code in its i-th corresponds to replace the i-th component by the value 0 in the original code: in our framework this corresponds to removing a sensor from position x_i; we refer to this representation since it has immediate correspondence to the position of the sensors.

In the next example we illustrate a technique to build a X-identifying set by iterative puncturing, which extends the case considered in [3].

Example. A simple example is reported in Fig. 1. The graph G is made of 5 nodes, and 6 edges, Set X is the set of mid edge points, i.e., given edge $e = v_{e_1}v_{e_2}$, $x_e = (v_{e_1} + v_{e_2})/2$. Two–thresholds function h is defined by thresholds $t_1 = 500$ m and $t_2 = 800$ m. We start from $\mathbb{C} = \{1, 2, 3, 4, 5\}$: for that code, it is easy to see that the distance of the code is $d = 1$. However, we can remove components 1 and 4, and consider the code which is obtained puncturing the original code on components 1 and 5. The X-identifying code $\mathbb{C} = \{2, 3, 5\}$ corresponds to the punctured code reported above.

III. Problem statement

In our context, minimizing the length of the codewords means deploying a minimal number of sensors.

X-Identifying Code: *"Given an Euclidean graph G, set X of points lying on edges of G, and characteristic function h, determine an identifying set* $\mathbb{C}$ *such that* $|\mathbb{C}|$ *is minimized"*

In general, the complexity of constructing optimal identifying codes for arbitrary graphs is known to be NP–complete [10]. Since $X = V$ is a particular case of our problem, it follows that the X-Identifying Code is at least as hard as the Identifying Code problem:

Theorem 1: The X-Identifying Code problem is NP-hard.

Following the same approach in [3], instead of looking for an optimal solution, a greedy algorithm to construct *irreducible* identifying codes: the deletion of any codeword results in a code that is no longer an identifying code; thus, the proposed algorithm always converges to a local minimum.

The minimum size of an X-identifying code can be lower bounded as $1 + \lceil \log_q |X| \rceil$ since that is the number of q-ary symbols required to encode an alphabet of $|X|$ elements; here $q = K + 1$.

IV. Iterative puncturing algorithm

The iterative puncturing procedure, reported in Fig. 2, is a greedy algorithm to reduce the length of a code: in our case, zeroing the i-th punctured component corresponds to eliminate sensor i. We obtain an iterative puncturing algorithm that produces an X-identifying code: at each step it preserves the initial code distance and it ensures that the number of codewords does not decrease.

```
$\mathbb{C}$=X-ID-CODE($G$,$X$, $\mathbf{a}$)

$X \leftarrow$ REDUCE($X$)
$\mathbb{C} \leftarrow V$, $\mathcal{C} \leftarrow \mathcal{C}(\mathbb{C})$
for $i = 1, \dots, N$,
    $\mathcal{C} \leftarrow \mathcal{C}^*_{[i]}$
    if $|\mathcal{C}| = |X|$ and $d^* \geq 1$
            $\mathbb{C} \leftarrow \mathbb{C} \setminus \{v_{[i]}\}$
endfor
return $\mathbb{C}$
```

Fig. 2: *The X-ID-CODE algorithm generates identifying code* $\mathbb{C}$ *for an arbitrary graph and set* X*.*

The pucturing of a code is characterized by the following.

Lemma 1: Let $\mathcal{C}$ be a $[n, m, d]$ code:

(i) If $d > 1$, $\mathcal{C}^*_i$ is a $[n-1, m, d^*]$ code where $d^* = d-1$ if $\mathcal{C}$ has two codewords that differ only in the i-th coordinate and $d = d^*$ otherwise

(ii) When $d = 1$, $\mathcal{C}^*$ is a $[N-1, M, 1]$ code if there are no two codewords that differ only in the i-th coordinate, whereas if there are r codewords that differ only for components i, $[N - 1, M - r, d^*]$, where $d^* > 1$.

Our aim here is to operate iterative puncturing such in a way to ensure that $M^* = M$ and $d^* \geq 1$ at each step. Denote $[\cdot]$ a permutation of $(1, \dots, N)$.

Theorem 2: At every step, the distance of the code generated by X-ID-CODE does not increase.

Proof: Since the X-ID-CODE algorithm is working with iterative puncturing, from Thm. 1, if $d(\mathcal{C}) > 1$, the distance $d(\mathcal{C})$ decreases at most of one unit at each step. The only case when the distance of the code can increase is when $d(\mathcal{C}) = 1$ and $r > 1$ codewords differ only for the puctured component i. However, the check $|\mathcal{C}| = |X|$ does not allow the removal of the codeword. ■

Remark 1: The above property states that either set V is a X-identifying set, or no X-identifying code exists. If it is so, X-ID-CODE maintains the code distance larger than 1 at each iteration, which in turn guarantees that the output set $\mathbb{C}$ is a X-identifying set. Counterexamples exist for the case when $\mathcal{C}(V)$ is not identifying, i.e., $d(\mathbb{C}(V)) = 0$. In our context, nevertheless, we aim at restricting the set of identified edges: whenever $d(\mathbb{C}(V)) = 0$, a simple procedure we identify all codewords of $\mathcal{C}(V)$ that have distance 0 among them and consider reduced set accordingly; we denote this procedure $X \leftarrow$REDUCE(X).

V. Incrementally Identifying Sets

The application of the previous concepts to WDSs requires the placement of noise detectors to *identify* the presence of leakages at positions corresponding to the set X. However, we should require that $\mathcal{C}$ is a covering for *any possible* position

```
$\mathbb{C}$, $\Delta\mathbb{C}$ = XCODE($G$,$X$,$\mathbf{a}$)

$\mathbb{C} \leftarrow V$, $\mathbb{D} \leftarrow V$, $\mathcal{C} \leftarrow \mathcal{C}(\mathbb{C})$
$X \leftarrow$ REDUCE($X$)
if $\mathcal{C}(\mathbb{C})$ is not edge-covering
    return $\mathbb{C} = \Delta C = \emptyset$ (no edge-covering code exists)
for $i = 1, \ldots, N$,
    if $\mathcal{C}(\mathbb{C})$ is edge-covering
        $\mathbb{D} \leftarrow \mathbb{D} \setminus v_{[i]}$
        $\mathcal{C} \leftarrow \mathcal{C}^*_{[i]}$
        if $|\mathcal{C}| = |X|$ and $d^* \geq 1$
            $\mathbb{C} \leftarrow \mathbb{C} \setminus v_{[i]}$
endfor
$\Delta\mathbb{D} \leftarrow \mathbb{C} \setminus \mathbb{D}$
return $\mathbb{D}$, $\Delta\mathbb{D}$
```

Fig. 3: *The XCODE algorithm training set X: output is edge covering set $\mathbb{D}$ and set of nodes $\Delta\mathbb{D}$ such that $\mathbb{C} = \mathbb{D} \cup \Delta\mathbb{D}$.*

of a leakage, i.e., for every $x_i \in e_j$ there exists $c_j \in \mathcal{C}$ such that $h(\ell_{ij}) \neq 0$: this corresponds to being able to *detect* the presence of a leakage in the waterpipe network. Thus, we require that $\mathbb{C}$ is a *edge-covering* set. A trivial necessary and sufficient condition for an *edge-covering* set to exist is that $h(d_{ij}/2) > 0$ for every link ij and we will assume that it is always satisfied. This requires that sensors must be capable of covering at least half the length of the longest pipe in the network (if this is not the case, additional nodes, i.e. new wells can be deployed to host sensors along such pipes). We can then propose the algorithm reported in Fig. 3 to obtain edge-covering set $\mathbb{D}$ that can be enriched such in a way to become an identifying set $\mathbb{C}$. In other words, $\Delta\mathbb{D} = \mathbb{C} \setminus \mathbb{D}$ can be used to incrementally extend $\mathbb{D}$ in order to obtain an X-identifying set.

Once the X-identifying set has been obtained, we aim at measuring the performance of a noisy detection, i.e., when the received codeword corresponding to a certain edge is y are affected by errors, namely a displacement with respect to the training set X. By means of minimum distance decoding, we would determine codewords of set $Y = \arg\min_{c \in \mathcal{C}}\{d(y, c)\}$ In this context, our aim is to minimize the number of candidate edges that can host a leakage: to this respect our approach is incremental in that the more elements of $\Delta\mathbb{D}$ we add to set $\mathbb{D}$, the smaller the set of candidate set of edges can be made.

VI. Numerical results

In this section we describe the simulation environment followed by the numerical evaluation. We assess the tradeoffs between number of nodes of the graph, i.e., the maximum number of positions that can host a sensor, the sensors coverage characteristics and number of sensor nodes actually deployed. Moreover we analyze the impact of the leakage position over pipes on our leakage detection algorithm. Finally, we consider the problem of providing a boostrap edge covering set capable of detecting leakages on the waterpipes and we show how the XCODE solution can perform progressively more accurate leakage identification by adding new sensors to the initial set.

A. Simulation Environment

Simulation have been performed using Matlab using three reference waterpipes network topologies: *grid*, *pruned grid*, and *pruned distorted grid*; pruning is operated using a random edge deletion probability of 0.5. The reference topologies used here are meant to mimic some properties of waterpipe networks which are typically planar graphs with degree not larger than 4 and median dgree 2. For each type of topology, we consider three instances characterized by a different number of nodes, in particular 10×10, 20×20, and 30×30 grids have been used.

For the *distorted grid* and the *pruned distorted grid* a library of 1000 random topologies has been used. Waterpipes' length was set to 300m for the *grid* and the *pruned grid* while it was uniformly distributed between 100m and 500m for the *pruned distorted grid* topology. Figure 4 reports an example of each of the three types of topologies adopted. All the results reported hereafter are the average of 1000 runs.

In our study we assume that the power of the noise generated by a leakage attenuates along pipes as $R(\ell) = 60\max(1-\frac{\ell}{1000}, 0)$: thresholds t_1 and t_2 correspond to reference power levels $R_{High} = R(t_1)$ and $R_{Low} = R(t_2)$.

B. Performance metrics.

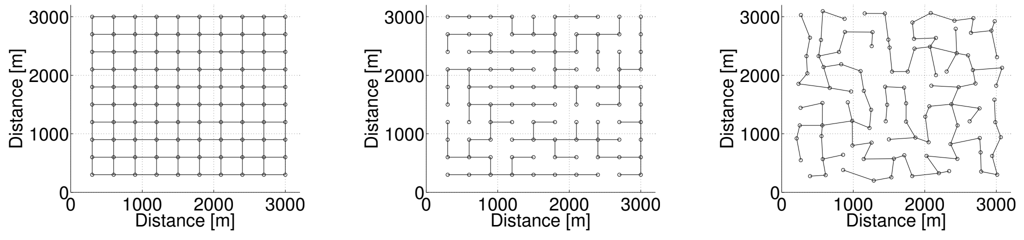
In order to measure the performances of our algorithm we used the following metrics:
a) *number of sensors*. The number of sensors required to identify a single leakage. Results are given for different values of the signature function's parameters h_1 and h_2. In particular we considered the following scenarios:

- $t_1 = t_2 = D$ where $D \in [300, 1000)$ m with increments of 100 meters;
- $t_2 = 833m$, t_1 such that $R(t_1) \in [11, 59)$ dB with increments of 3 dB.
- $t_1 = 500m$, $t_2 = 833m$, which corresponds to an $R_{High} = 30$ dB and $R_{Low} = 10$ dB;

b) *sensitivity to leakage position:* XCODE algorithm is trained by a certain reference X set: in our experiments the set of mid-edge points. Thus we evaluate the robustness of the algorithm over a fine grid of position to be identified; we simulated the presence of a *single* leakage on each edge and by computing the number of edges that are thus classified as faulty. Leakages are placed over 100 positions uniformly spaced over each edge. Results are given for different leakage placements relative to the reference training case, i.e., the center of the edge.

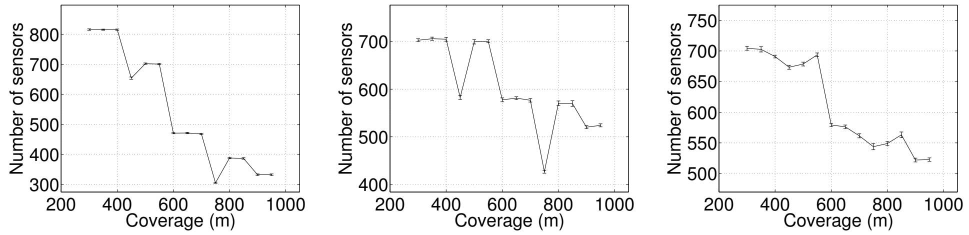
C. Results

Figure 5 reports on the average number of sensors required to identify leakages, i.e., to produce a code of distance 1, for various types of topologies; it is evaluated versus the sensor coverage. As seen there, the number of sensors required to identify a leakage depends on the number of nodes, i.e. the junctions, and on the sensor coverage. In particular, we find



(a) 10×10 *Grid.* **(b)** 10×10 *Pruned grid.* **(c)** 10×10 *Pruned distorted grid.*

Fig. 4: *Reference waterpipes topologies used for our numerical evaluation.*



(a) 30×30 *Grid.* **(b)** 30×30 *Pruned Grid.* **(c)** 30×30 *Pruned Distorted Grid.*

Fig. 5: *Average number sensors required to identify leakages in various topologies Vs. sensor coverage.*

that using sensors with a coverage equal to the double of the average pipe's length minimizes the number of sensors required to identify leakages on the network. This can be ascribed to the fact that using a two threshold characteristic, such configuration tends to increase the diversity in the signature corresponding to the training set X.

Figure 6 reports the average number sensors required to identify leakages in various topologies for different sensitivity values. In particular we kept the R_{Low} sensitivity, i.e. the strength below which every signal is classified as background noise, fixed to 10 dB which corresponds to a coverage distance of 833 meters, while we changed the R_{High} sensitivity between 11 and 59 dB which corresponds to a coverage distance of, respectively, 816 and 16 meters. The x-axis on the picture reports directly the sensor coverage.

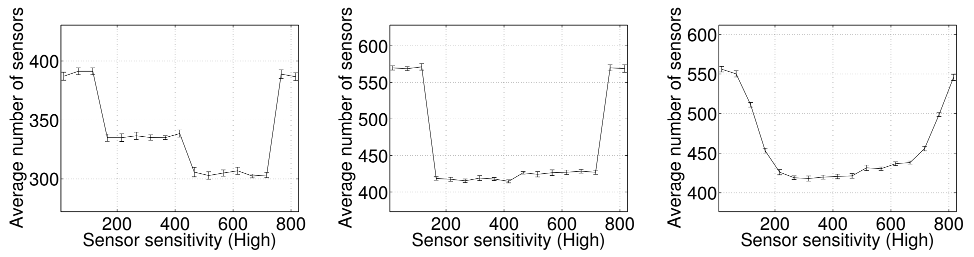
We observe that, on the *Grid* topology the number of sensors required to identify leakages on the network is maximum when the R_{High} is either very high, i.e. low coverage, or when it is approaching the R_{Low} value, i.e. $h_1 = h_2$ which means that a the generated code becomes binary instead of ternary. This behavior is easily explained by the the syndromes corresponding to every leakage have reduced diversity in those cases, i.e., sensor coverage is too large and too small, respectively, compared to the network typical edge length. No particular pattern can be found for the *Pruned grid* and for the *Pruned distorted grid* topologies.

Figure 7 reports the average number of edges identified as faulty for randomly occurring *single* leakages. Notice that, the identifying code has been computed for leakages placed in the middle of each edge. Each point in the graph reports the average number of edges that are classified as faulty when a *single* leakage is placed at different position on the edge. The 95% confidence interval is reported as error bar. Again, a value close to the average length of the edges seems to deliver the best performance: in the ideal case (0.5) only one edge is classified as faulty. However, it is worth noticing that, even in the worst case scenario, i.e. when a leakage is placed at the endpoints of an edge, we can significantly reduce the number of edges that are potentially hosts for a leakage.

D. Leakage coverage with incremental identification

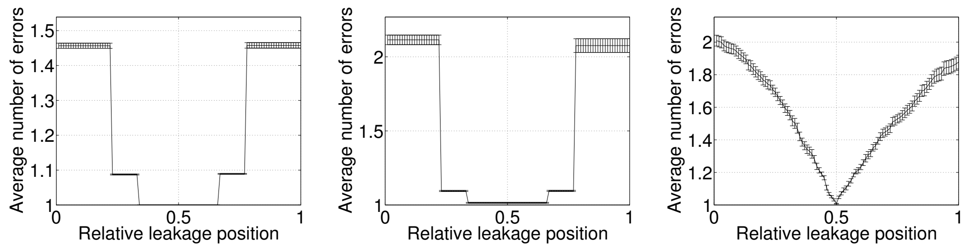
In this section we describe the performance of the covering algorithm XCODE. It is worth stressing that X-ID-CODE finds first an identifying set $\mathbb{C}$, i.e. a set of sensors inducing a code with distance 1 associated to the set X; then XCODE prunes the identifying set $\mathbb{C}$ to obtain edge covering set $\mathbb{D}$ such that a point lying on any edge in the network is covered by at least one sensor. Clearly, the edge-covering set $\mathbb{D}$ is not guaranteed to uniquely identify set X. Table II reports on the number of sensors required to identify leakages and to just cover all the edges for various topologies; 95% confidence intervals are reported in brackets. As seen there the covering set $\mathbb{D} = \mathbb{C} - \Delta\mathbb{C}$, which just cover all the edges in the network has roughly half the sensors required to identify leakages, i.e., $2|\Delta\mathbb{C}| \simeq |\mathbb{C}|$. Table II reports the fraction of the edges that are classified as affected by a leakage when only the covering set is used. As seen there, the edge-covering set can identify the leakage only in 60% of cases; however, we observe that only 10% of errors ascribe the fault to more than two waterpipes.

Figure 8 reports on the average number of errors, i.e. the number of edges misclassified as hosting a leakage when an actual leakage occurs in a neighboring edge, when an increasing number of sensors belonging to $\Delta\mathbb{D}$ is added to the bootstrap deployment $\mathbb{D}$. As it can be seen, a *linear* relationship exists between the number of sensor deployed and the detection accuracy. Such behavior is beneficial for the water provider that can first bootstrap a large scale effort to sanitize the network using the set $\mathbb{D}$ and then incrementally upgrade system to obtain single leakage detection once all the $\Delta\mathbb{D}$ positions have been covered.



(**a**) 30 × 30 *Grid.* (**b**) 30 × 30 *Pruned Grid.* (**c**) 30 × 30 *Pruned Distorted Grid.*

Fig. 6: *Average number sensors required to identify leakages in various topologies Vs. sensor sensitivity* h_1.



(**a**) 30 × 30 *Grid.* (**b**) 30 × 30 *Pruned Grid.* (**c**) 30 × 30 *Pruned Distorted Grid.*

Fig. 7: *Average number of edges identified as faulty Vs relative position of a leakage on the edges.*

Topology	Identifying	Covering	Number of identifyed edges		
			1	2	3+
Grid 10x10	36.1 (1.3)	21 (0.72)	68	22	8
Grid 20x20	137.4 (2.3)	81.7 (1.5)	71	23	5
Grid 30x30	306.5 (3)	184.4 (2.1)	76	19	4
Pruned Grid 10x10	44.9 (0.94)	26.2 (1)	59	24	16
Pruned Grid 20x20	191.9 (3.1)	117.6 (3)	68	19	11
Pruned Grid 30x30	425.3 (3.2)	261.2 (3)	68	21	10
Dist. Pruned Grid 10x10	44.4 (1.2)	25.7 (0.66)	61	24	13
Dist. Pruned Grid 20x20	188.6 (3.5)	116.6 (2.3)	70	20	8
Dist. Pruned Grid 30x30	423.2 (2.4)	263.8 (2.7)	71	20	8

TABLE II: *Identifying Vs. Covering performance: Number of sensors and detection errors. Confidence intervals in brackets.*

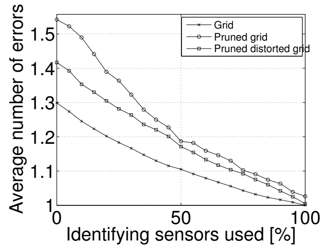


Fig. 8: *Number of links identified Vs increasing number of sensors.*

VII. Conclusions

In this paper we presented a framework for the placement of sensors on a lattice type of infrastructure for monitoring purposes: our reference case is that of waterpipe networks. Target is to deploy a minimum number of acoustic sensors such in a way to detect the presence of leakages and determine which pipe to repair. Our solution enables a two step deployment strategy, where one would perform a network covering deployment first, in order to sanitize the network, and then improve the resolution by incrementally deploying the complete the set of sensors to form an identifying set. In future work we will include in our model the presence of heterogeneous materials along pipes (metal, plastic) and we will validate the results on the production waterpipe network of Dolomiti Energia S.p.A. in Trento.

Acknowledgments

The authors would like to acknowledge the fruitful discussions with Matteo Frisinghelli of Dolomiti Energia on the leakage detection problem.

References

[1] V. J. Garcia and E. Cabrera, “The minimum night flow method revisited,” in *Annual Water Distribution Systems Analysis Symposium*, 2006.

[2] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin, “A new class of codes for identification of vertices in graphs,” *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 599–611, March 1998.

[3] S. Ray, R. Ungrangsi, F. D. Pellegrini, A. Trachtenberg, and D. Starobinski, “Robust location detection in emergency sensor networks,” in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, 2003.

[4] B. Wu, P.-H. Ho, and K. L. Yeung, “Monitoring trail: A new paradigm for fast link failure localization in wdm mesh networks,” in *Proc. of IEEE GLOBECOM*, Zurich, Switzerland, 2008.

[5] J. Tapolcai, L. Rónyai, and P.-H. Ho, “Optimal solutions for single fault localization in two dimensional lattice networks,” in *Proc. of IEEE INFOCOM*, San Diego, California, USA, 2010.

[6] S. S. Ahuja, S. Ramasubramanian, and M. M. Krunz, “Single-link failure detection in all-optical networks using monitoring cycles and paths,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1080–1093, Aug. 2009.

[7] H. Zeng, C. Huang, and A. Vukovic, “A novel fault detection and localization scheme for mesh all-optical networks based on monitoring-cycles,” *Photonic Network Communications*, vol. 11, pp. 277–286, 2006.

[8] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber, and J.-P. Watson, “Sensor Placement in Municipal Water Networks with Temporal Integer Programming Models,” *Journal of Water Resources Planning and Management*, vol. 132, no. 4, pp. 218+, 2006.

[9] Z. Y. Wu and T. Walski, “Multi-objective optimization of sensor placement in water distribution systems,” in *Proc. of ASCE Water Distribution Systems Analysis Symposium*, 2006.

[10] N. S. V. Rao, “Computational complexity issues in operative diagnosis of Graph-Based systems,” *IEEE Transactions on Computers*, vol. 42, no. 4, pp. 447–457, April 1993.